Modular and Distributed Verification of SysML Activity Diagrams

Messaoud Rahim, Ahmed Hammad, Malika Ioualalen

▶ **To cite this version:**

Messaoud Rahim, Ahmed Hammad, Malika Ioualalen. Modular and Distributed Verification of SysML Activity Diagrams. MODELSWARD 2013, 1st Int. Conf. on Model-Driven Engineering and Software Development, Jan 2013, Spain. pp.202 - 205. hal-00935748

HAL Id: hal-00935748
https://hal.science/hal-00935748

Submitted on 24 Jan 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Modular and Distributed Verification of SysML Activity Diagrams

Messaoud Rahim[1], Ahmed Hammad[2] and Malika Ioualalen[3]
[1]*Sciences and Technology Faculty, Yahia Fares University, Medea, Algeria*
[2]*Institut FEMTO-ST , UMR CNRS 6174, Besancon, France*
[3]*LSI, Computer Science department, USTHB, Algiers, Algeria*
{*rahim_mes@yahoo.fr, ahmed.hammad@femto-st.fr, mioualalen@usthb.dz*}

Keywords: SysML, Activity diagram, Places bordered Petri nets, Distributed Model-Checking.

Abstract: Model-based development for complex system design has been used to support the increase of systems complexity. SysML is a modeling language that allows a system description with various integrated diagrams, but SysML lacks formality for the requirement verification. Translating SysML-based specification into Petri nets allows to enable rigorous system analysis. However, for complex systems, we have to deal with the state space explosion problem. In this paper, we propose new approach to allow a modular and distributed verification of SysML Activity Diagram basing on the derived Petri net.

1 Introduction

The System Modeling Language (SysML) is UML profile that can be used to specify graphically all aspects of complex systems(Friedenthal et al., 2008). Nevertheless, despite the various advantages of SysML, it remains a semi-formal language without possibilities of formally verifying the models described by it.
Industrial safety-related standards strongly recommend the use of formal methods to validate critical systems. For that purposes, it is needed to use SysML in conjunction with formal method to provide formal verification of the specified system. Several approaches based on mapping SysML behavioural diagrams to Petri nets have been proposed (Carneiro et al., 2008; Andrade et al., 2009; Linhares et al., 2007). The aim of these approaches was to provide a way to verify the specified system with a Model checking technique. However, in the case of complex systems, we have to deal with the state space explosion problem to analyse the resulting Petri net. A way to overcome the state space explosion is the use of modular analysis. Another way that had gained interest, in the recent years is the use of distributed processing (Kristensen and Petrucci, 2004; Saad et al., 2010; Barnat and Rockai, 2008).
In this paper, we propose a global approach for performing a modular and distributed verification of the SysML activity diagram. Basing on composite activities, we derive places-bordered Petri net module for each activity. The verification of the system can concern only one simple activity or the global SysML activity diagram. For the second case, and in order to deal with the state space explosion problem, we propose to adapt the distributed verification process using a cluster of computing nodes(Boukala and Petrucci, 2011; Abid and Zouari, 2007) for verifying the derived modular Petri net. For mapping a SysML activity diagram into places-bordered Petri net, we propose a translation rule for the call behavior action. The translation of the other basic SysML activity constructs is inspired from previous works.

The rest of this paper is organized as follows: in Section 2, we discuss related works. In Section 3, we present the SysML activity diagram. In Section 4, we give a definitions of places bordered Petri net. In section 5, we present the mapping technique. We present the modular and distributed verification process in Section 6. Finally, in Section 7, we conclude and we outline some ideas for future works.

2 Related Works

The most proposed approaches concerning the formal specification of SysML diagrams have used Petri net models due to their expressiveness and formality (Linhares et al., 2007; Carneiro et al., 2008; Andrade et al., 2009). To our knowledges, this work is the first that considers the composite structure of the

SysML activity diagram for a verification purposes. The composite and modular verification approaches aim to take benefit from some information about the components of the system and the way they communicate. Modular Petri nets as presented in (Christensen and Petrucci, 2000) allow designers to specify a system as communicating modules. Modules communicate using shared transitions or fusion places. The work presented in (Valmari, 1994) proposes a compositional verification method for Petri net composed of place bordered subnets. The verification approach used in this work is based on Model checking technique. However, for complex system, we have to overcome the state space explosion problem. Several recent approaches use distributed processing environments to extend the size of the state space to be constructed(Boukala and Petrucci, 2011). This work proposes to adapt these approaches combined with a modular analysis to verify a SysML activity diagrams.

3 The SysML Activity diagram

In SysML(**?**), an activity is a formalism for describing behaviour that specifies the transformation of inputs to outputs through a controlled sequence of actions. The basic constructs of an activity are actions and control nodes as illustrated in Figure 1. Actions are the building blocks of activities, each action can accept inputs and produces outputs, called tokens. These tokens can correspond to anything that flows such as information or physical item (e.g., water, signal). Control nodes include fork, join, decision, merge, initial, activity final, and flow final. A call behavior action permits to invoke an activity when it starts, and passes the tokens from its input pins to the input parameter nodes of the invoked activity.

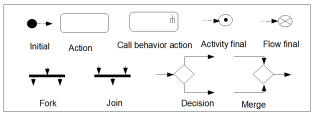

Figure 1: Activity diagram basic constructs

4 Places-bordered Petri nets

Formally a Petri net is (Valmari, 1994):

Definition 1. *A Petri net is triplet: $PN = (P,T,W)$. Where: P is finite set of places, T is finite set of transitions, $(P \cap T = \emptyset)$ and $W : (PXT) \cup (TXP) \rightarrow N$ is a weight function, $W(p,t)$ (resp. $W(t,p)$) gives the weight of the arc from p to t (resp. from t to p).*

Usually, an initial marking is associated with the Petri net:

Definition 2. *A marked Petri net (PN, M_0), is a Petri net PN with an initial marking $M_0 : P \rightarrow N$. The initial marking of a place $p \in P$ is $M_0(p)$.*

To compose a large Petri net from smaller pieces, we define a places-bordered Petri (Valmari, 1994). A places-bordered Petri net modules interface with each other via common places, called border places.

Definition 3. *A places-bordered Petri net module is the 4-tuple $NC = (P,T,W,B)$. where : P, T, and W are as in a Petri net, $B \subseteq P$ is the set of border places.*

5 The mapping technique

Basing on the previous works that propose a mapping of UML and SysML activity diagram to Petri nets (N. Yang and Qian, 2010; Andrade et al., 2009; Staines, 2008), our technique defines a mapping for the call behavior actions and propose to map the SysML activity diagram to modular Petri net with border places. The mapping we propose is activity based decomposition. The decomposition is guided by the call behavior actions which permits to facilitate the mapping of a SysML activity diagram even it includes several composite activities. The Petri net derived from the SysML activity diagram is a set of places-bordered Petri net modules, each one represents an activity instance.

5.1 Mapping initial and final nodes

Initial node represents the start point of an activity. As illustrated in figure 2, to map the initial node, we use one transition (t_in_Act) with one input place and two output places. The input place (en_Act) is used to enable the execution of the activity. The first output place (on_Act) is used to indicate that we are executing the activity and the second output place (Ctl_out) is used to represent the control flow.
Activity final represents the end point of an activity. As illustrated in figure2, to map final node, we use one transition (t_out_Act) with two input places and one output place. The first input place (on_Act) represents that we are executing the activity and the second input place (Ctl_in) represents the output flow enabling the termination of the activity. The output place (end_Act) is used to indicate that the activity is terminated.

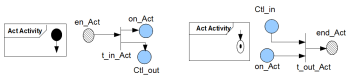

Figure 2: Mapping initial and final activity node

5.2 Mapping actions and object flows

As illustrated in the figure 3, for mapping an action with control and data flow into Petri net, places are used to represent the input and the output flows $(in\_A, Ctl\_in\_A, out\_A, Ctl\_out\_A,)$ and one transition $(Exec\_A)$ is used to represent the action.
For mapping an object flow that connects output pin of one action A to the input pin of another action B, we fusion the place (Out_A) that represents the output pin of the action A with the place (In_B) which represents the input pin of the action B.

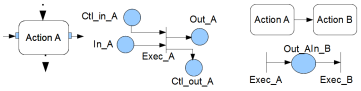

Figure 3: Mapping simple action and Object flows between actions

5.3 Mapping routing object flows

For mapping a fork node 4 we use a transition t_A_fork that represents the split operation with out_A as input place and in_B with in_C as output places. For mapping a join node 4 we use a transition t_AB_join that represents the synchronisation between out_A and out_B as input places and in_C as output place.

Figure 4: Mapping join and fork nodes

5.4 Mapping call behavior action

To map A call behavior action we consider that the invoked activity is already mapped into places bordered Petri net module. As presented in Figure 5, the mapping of a call behavior action A that invokes an activity Act with one input and one output flow is places bordered Petri net module which has in_act, en_Act, out_Act and end_Act as border places with the Petri net module that represents the calling activity. The transition t_Abact is used to pass all input flows of the call behavior action to the invoked activity and to enable its execution. When the called activity terminates, we use the transition t_Aeact to pass all output flows of the invoked activity to the call behavior action.

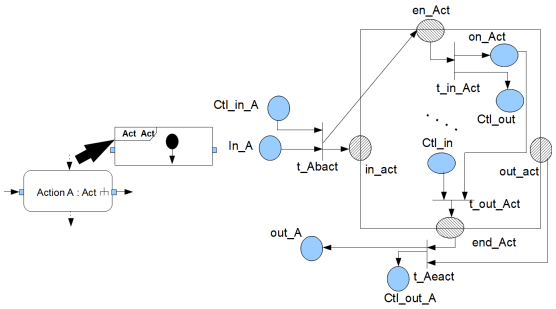

Figure 5: Mapping call behavior action

6 The modular and distributed verification process

As described in the mapping technique, the resulting Petri net is set of place bordered Petri net modules. Modular verification is enabled by the fact that each Petri net module specifies the behavior of an activity. A simple activity can be verified by using only its related Petri net module. For verifying a composite activity we have to use its related Petri net module and all the Petri net modules corresponding to their call behavior actions. A modular and distributed verification can be used to perform the analysis of a complex composite activities. The main step in the model-checking is the construction of the state space. In order to construct the state space of the Petri net derived from the SysML activity diagram we decompose the state space construction problem into a number of distributed tasks. The decomposition is guided by the SysML activities. The approach we propose is parallel objects based (Kale and Zheng, 2009). For constructing the state space, we use one main task to initialize the state space construction process, and a set of parallel tasks which we call activity tasks to explore and store the state space. Activity tasks are used to encode a place bordered Petri net modules. Each Petri net module is assigned to an activity task which explores independently the internal states of the module. Activity tasks encapsulate informations about transitions and border places. They perform the exploration of a given state, the storing of internal successors state, seek for previously explored states and invoke the storing of external states.

Each activity task encapsulates a hash table that is used to store a fragments of the state space. From a logical point of view, we consider all the processing nodes as a unique computing node. The activity tasks are viewed as an arrays of parallel Tasks. Physically, all the tasks are mapped over the physical nodes. The mapping is done when we create the tasks. The Figure 6 presents an example using four (4) computing nodes

to illustrate the logical and the physical architecture of the distributed application.

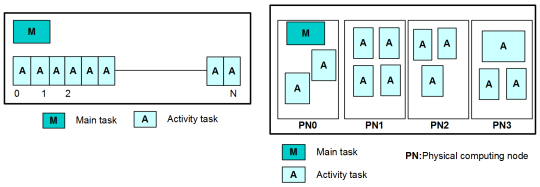

Figure 6: Logical and physical view of the application architecture

The state space is the basic model on which most verifications are built. The constructed state space is modular. It can be used to verify behavioural properties in the hole activity diagram or just on some activities. The properties to verify can be basic such as reachability, deadlocks, liveness and home state. We can adopt the approach presented in (Boukala and Petrucci, 2011) to verify such properties in the distributed and modular state space. Functional properties have to be extracted and translated from the SysML requirements diagram to temporal logic such as LTL and CTL. Various works have been proposed to verify LTL and CTL formulas in distributed and modular state space (Christensen and Petrucci, 2000; Latvala and Makela, 2004).

7 Conclusion

The paper presents a modular and distributed verification approach for formally verifying complex systems described by SysML activity diagrams. A technique for mapping the SysML activities to Petri net have been proposed. The mapping is guided by the call behavior actions. The modular verification is enabled by analysing each activity using its related Petri net module. For enabling the verification of complex and composite activities a modular and distributed verification technique have been proposed to overcome the state space explosion problem. As future works, it is important to consider the process of extracting properties as temporal logic formulas from the SysML requirement diagram to complete the approach presented in this paper.

REFERENCES

Abid, C. A. and Zouari, B. (2007). A distributed verification approach for modular petri nets. In *on Proceedings of the 2007 summer computer simulation conference*, pages 681–690.

Andrade, E., Macie, P., Callou, G., and Nogueira, B. (2009). A methodology for mapping sysml activity diagram to time petri net for requirement validation of embedded real-time systems with energy constraints. In *Third International Conference on Digital Society, ICDS'09*, pages 266–271.

Barnat, J. and Rockai, P. (2008). Shared hash tables in parallel model checking. In *Electronic Notes in Theoretical Computer Science 198(1), proceedings of the 6th International Workshop on PDMC 2007*.

Boukala, M. and Petrucci, L. (2011). Distributed verification of modular systems. In *in Proceedings of CompoNet and SUMo*.

Carneiro, E., Maciel, E., Callou, P., and Tavares, G. (2008). Mapping sysml state machine diagram to time petri net for analysis and verification of embedded real-time systems with energy constraints. In *International Conference On Advances in Electronics and Microelectronics ENICS '08*, pages 1–6.

Christensen, S. and Petrucci, L. (2000). Modular analysis of petri nets. In *The Computer Distributed Verification of Modular Systems Journal 43*, pages 224–242.

Friedenthal, S., Moore, A., and Steiner, R. (2008). Omg systems modeling language (omg sysml) tutorial. In *Published and used by INCOSE*.

Kale, L. V. and Zheng, G. (2009). Charm++ and ampi: Adaptive runtime strategies via migratable objects. In *In M. Parashar, editor, Advanced Computational Infrastructures for Parallel and Distributed Applications*, page 265282. Wiley-Interscience.

Kristensen, S. and Petrucci, L. (2004). An approach to distributed state space exploration for coloured petri nets. In *In Proc. 25th Int. Conf. Application and Theory of Petri Nets (ICATPN2004), Bologna, Italy*, pages 474–483.

Latvala, T. and Makela, M. (2004). Ltl model checking for modular petri nets. In *In in proc. of ICATPN'04*, pages 298–311.

Linhares, M.-V., de Oliveira, R.-S., Farines, J.-M., and Vernadat, F. (2007). Introducing the modeling and verification process in sysml. In *12th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA'2007*, pages 344–351.

N. Yang, H. Yu, H. S. and Qian, Z. (2010). Mapping uml activity diagrams to analyzable petri net models. In *Proc. of the 2010 10th Int. Conf. on Quality Software, IEEE, Zhangjiajie*, pages 369–372.

Saad, R., Zilio, S., and Berthomieu, B. (2010). A general lockfree algorithm for parallel state space construction. In *proceedings of the 9th International Workshop on Parallel and Distributed Methods in verifiCation*, pages 8–16.

Staines, T. S. (2008). Intuitive mapping of uml 2 activity diagrams into fundamental modeling concept petri net diagrams and colored petri nets. In *15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, ECBS 2008*, pages 191–200.

Valmari, A. (1994). Compositional analysis with place-bordered subnets. In *Proceedings of the 15th International Conference on Application and Theory of Petri Nets*, pages 531–547.